Homework 6

CMSY-199, Spring 2013

The source code for this assignment must be submitted electronically using the Canvas course website prior to the start of class on Monday, May 6.

1. Write a class called `Calculator` which *is* a `JFrame` from the `javax.swing` package.
2. Make the `Calculator` class a Java application by adding a `main` method with a single line of code that creates an instance of the `Calculator` class named `c`
3. In addition to the `main` method, the `Calculator` class *has* 22 member variables, a no-argument constructor, and a method called `makeButton` which takes a `String` argument and returns a `JButton`. The 22 member variables consist of:
   (a) A `JTextField` for the display.
   (b) A `JPanel` container to hold the buttons.
   (c) Twenty `JButton` objects for:
      i. The numbers 0-9.
      ii. The arithmetic operators plus, minus, times, and divided by.
      iii. The clear display and the plus/minus toggle operation.
      iv. The equals operation and the decimal point.
      v. The store to memory and the recall from memory operations.
4. Write code in the no-argument constructor to:
   (a) Call the constructor of the superclass with the argument `"Calculator"`
   (b) Set the default close operation to exit on close.
   (c) Set the width and height to 300 pixels.
   (d) Set the resizable property to false.
   (e) Initialize the display to `0`
   (f) Set the horizontal alignment of the display to the right.
   (g) Set the font of the display to 48 point Dialog.
   (h) Set the focusable property of the display to false.
   (i) Create a `GridLayout` object with 5 rows and 4 columns.
   (j) Set the horizontal and vertical gaps of the layout to 5 pixels.
   (k) Initialize the `JPanel` which will hold the buttons with the layout.

(l) Call the `makeButton` method to initialize the twenty buttons as shown in the figure below.

(m) Add the display to the calculator at the north field of a `BorderLayout`

(n) Add the buttons to the calculator at the center field of a `BorderLayout`

(o) Set the visible property of the calculator to true.

5. Write code in the `makeButton` method to:

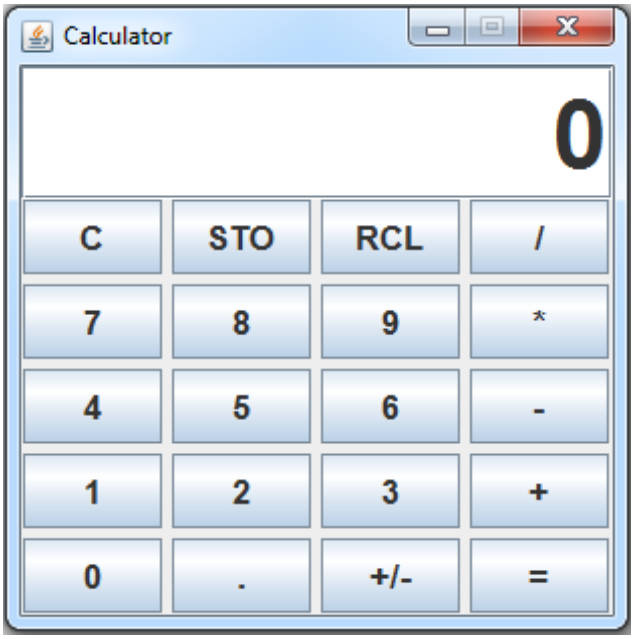
   (a) Create a `JButton` with the text from the `String` argument.

   (b) Set the font of the button to 17 point Dialog.

   (c) Set the focusable property of the button to false.

   (d) Add the button to the `JPanel` containing the buttons.

   (e) Return the button to the caller.



6. Make the `Calculator` class implement an `ActionListener` interface from the `java.awt.event` package.

7. The `ActionListener` interface requires that you implement a method called `actionPerformed` which takes an argument of type `ActionEvent` and has a return type of void. Write the `actionPerformed` method similar to the code below where a particular method is called depending upon the source of the `ActionEvent`.

8. Add an `ActionListener` to each button as it is created in the `makeButton` method.

9. Add five new member variables to the `Calculator` class:
   (a) Two doubles to represent the left and right operands
   (b) A String to represent the operator
   (c) A boolean to indicate that the left operand has been input and the display must be cleared when input of the right operand begins
   (d) A double for the value which has been stored in memory
10. Write the event-handling methods to take the appropriate action based on the source of the event and the state of the calculator at the time of the event.
11. The starting state of the calculator and the state after pressing the clear button should be as follows:
   (a) The display shows 0
   (b) The value of the left and right operands are 0
   (c) The operator is set to the empty string
   (d) The starting value in memory is 0 and not cleared when the clear button is pressed

```
public void actionPerformed(ActionEvent e)
{
   JButton source = (JButton) e.getSource();
   if (source == clear)
   {
      clearCalculator();
   }
   else if(source == store)
   {
     storeValue();
   }
   else if(source == recall)
   {
      recallValue();
   }
   else if(source == plus  || source == minus ||
           source == times || source == dividedBy)
   {
      setOperation(source);
   }
   else if (source == equals)
   {
      evaluateExpression();
   }
   else if (source == plusMinus)
   {
      togglePlusMinus();
   }
   else if (source == point)
   {
      addDecimalPoint();
   }
   else // has to be a number
   {
      addDigit(source);
   }
}
```